MSBD5003 Project Report

Purchase Analysis & Customer Profiling with Spark

CHEN I Chieh, YIN Zhuohao

MSBD5003: Big Data Computing

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

May 29, 2024

Contents

1 Introduction

In today's data-driven world, businesses are increasingly relying on data analysis to gain valuable insights and make informed decisions. One area where data-driven analysis can bring significant value is understanding customer profiles and purchasing behaviors. By mining large datasets, businesses can reveal patterns, trends, and correlations that can help them tailor their products and services to meet customer needs more effectively.

1.1 Task Description

Although detailed customer profiles and transaction data are rarely released by companies, we make use of a Kaggle dataset to develop a toy project that can be seamlessly scaled up due to the eminent scalability of distributed computing systems. The dataset contains rich information about customers of a supermarket over a period of several years. First, it includes demographic information about the customers, such as education, year of birth, etc. Second, it records the amount of several types of products that a customer purchased in the last 2 years, such as wine, fruits, and meat. Third, it records customers reaction to promotions. Last but not least, it specifies the number of times of the purchasing method, including web, catalogue, and store. A detailed specification of the features can be found in Table 1.

Among the wide range of features, we leave out the ones related to promotions as they are poorly documented and we are uncertain about their true meanings. Making use of the other features, we aim to perform detailed data mining tasks to extract meaningful insights. Specifically, we seek to answer two key questions: **(1)** How is the overall expense related to the demographic information of the customer? **(2)** Are there clusters of customers? If yes, what are the characteristics of each cluster?

1.2 Big Data Technologies

Cloud computing provides the infrastructure and resources necessary to handle large-scale data processing tasks. It offers scalability, flexibility, and cost-effectiveness, allowing businesses to leverage massive computational power without the need for expensive hardware investments. In this project, we primarily use PySpark [4] to implement our system. In particular, Spark SQL [6] is used to load, pre-process, and manipulate tabular data. Resilient Distributed Dataset (RDD) [5] is used to perform more user-specified transformations on the data. The Machine Learning Library (MLlib) [2] is used to perform machine learning algorithms for data mining purposes, such as linear regression and K-means clustering. Lastly, Spark Streaming is used for dynamic analysis of streaming data. Apart from Spark, we also utilize Pandas [3] for viewing dataframes, and Matplotlib [1] for producing plots.

2 Data Pre-processing

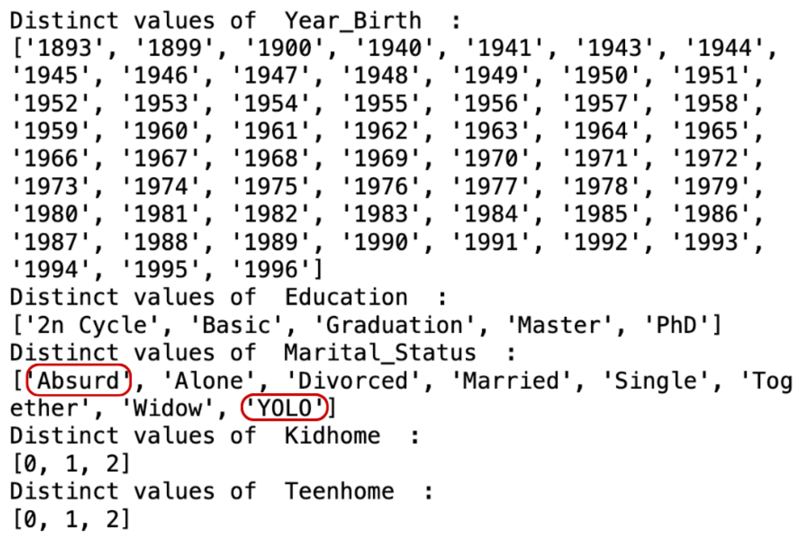
The first step of data pre-processing is data cleansing. For categorical features, we perform a sanity check on their values and see if all of them make sense. The program

Feature	Presumed Specification
ID	Customer's unique identifier
Year_Birth	Customer's birth year
Education	Customer's education level
Marital_Status	Customer's marital status
Income	Customer's yearly household income
Kidhome	Number of children in customer's household
Teenhome	Number of teenagers in customer's household
Dt_Customer	Date of customer's enrollment with the company
Recency	Number of days since customer's last purchase
Complain	1 if the customer complained in the last 2 years, 0 otherwise
MntWines	Amount spent on wine in last 2 years
MntFruits	Amount spent on fruits in last 2 years
MntMeatProducts	Amount spent on meat in last 2 years
MntFishProducts	Amount spent on fish in last 2 years
MntSweetProducts	Amount spent on sweets in last 2 years
MntGoldProds	Amount spent on gold in last 2 years
NumDealsPurchases	Number of purchases made with a discount
AcceptedCmp1	1 if customer accepted the offer in the 1st campaign, 0 otherwise
AcceptedCmp2	1 if customer accepted the offer in the 2nd campaign, 0 otherwise
AcceptedCmp3	1 if customer accepted the offer in the 3rd campaign, 0 otherwise
AcceptedCmp4	1 if customer accepted the offer in the 4th campaign, 0 otherwise
AcceptedCmp5	1 if customer accepted the offer in the 5th campaign, 0 otherwise
Response	1 if customer accepted the offer in the last campaign, 0 otherwise
NumWebPurchases	Number of purchases made through the company's website
NumCatalogPurchases	Number of purchases made using a catalogue
NumStorePurchases	Number of purchases made directly in stores
NumWebVisitsMonth	Number of visits to company's website in the last month

Table 1: Specifications of the features in the dataset.

output is shown in Figure 1. To clean the data, we remove the entries with unreasonable values and also merge the values that represent the same meaning. For example, marital status is categorized into either ”Partner” or ”Alone”. Additionally, we find 24 missing values in the *Income* column and we decide to use median imputation to fill the missing values by the median income.

Next, we perform feature engineering to the existing features in the following ways. First, we consider some features indirect and thus convert them to more intuitive ones. For instance, we create a new feature named *Age* by taking the difference between the current year and the year of birth. Moreover, we derive the *daysSinceEnroll* column by calculating the total days since the customer's enrollment. Second, we aggregate some features into one. For example, we define a new column *Children*, which reflects the total number of children at home by summing up *Kidhome* and *Teenhome*. Apart from that, we define the *Expense* column to be the sum amount of all types of products, which is later used as the response variable in linear regression. Third, we create new features that we consider insightful, such as the boolean column *isParent*, which is `True` if the number of children at home is at least 1 and `False` otherwise. Finally, we use a pairplot to both examine the data and detect outliers. As shown in Figure 2, on the diagonal are the histograms of the visualized features while the others are scatter plots with respect to 2 features. Among the scatters, we can clearly spot several outliers in the *Age* and *Income* columns. Thus, we filter our data to eliminate the entries whose age or income is exceptionally high.



```
Distinct values of  Year_Birth  :
['1893', '1899', '1900', '1940', '1941', '1943', '1944',
'1945', '1946', '1947', '1948', '1949', '1950', '1951',
'1952', '1953', '1954', '1955', '1956', '1957', '1958',
'1959', '1960', '1961', '1962', '1963', '1964', '1965',
'1966', '1967', '1968', '1969', '1970', '1971', '1972',
'1973', '1974', '1975', '1976', '1977', '1978', '1979',
'1980', '1981', '1982', '1983', '1984', '1985', '1986',
'1987', '1988', '1989', '1990', '1991', '1992', '1993',
'1994', '1995', '1996']
Distinct values of  Education  :
['2n Cycle', 'Basic', 'Graduation', 'Master', 'PhD']
Distinct values of  Marital_Status  :
['Absurd', 'Alone', 'Divorced', 'Married', 'Single', 'Tog
ether', 'Widow', 'YOLO']
Distinct values of  Kidhome  :
[0, 1, 2]
Distinct values of  Teenhome  :
[0, 1, 2]
```

Figure 1: Sanity check outputs. Unreasonable values, such as ”YOLO” and ”absurd”, are in the *Marital_Status* column.

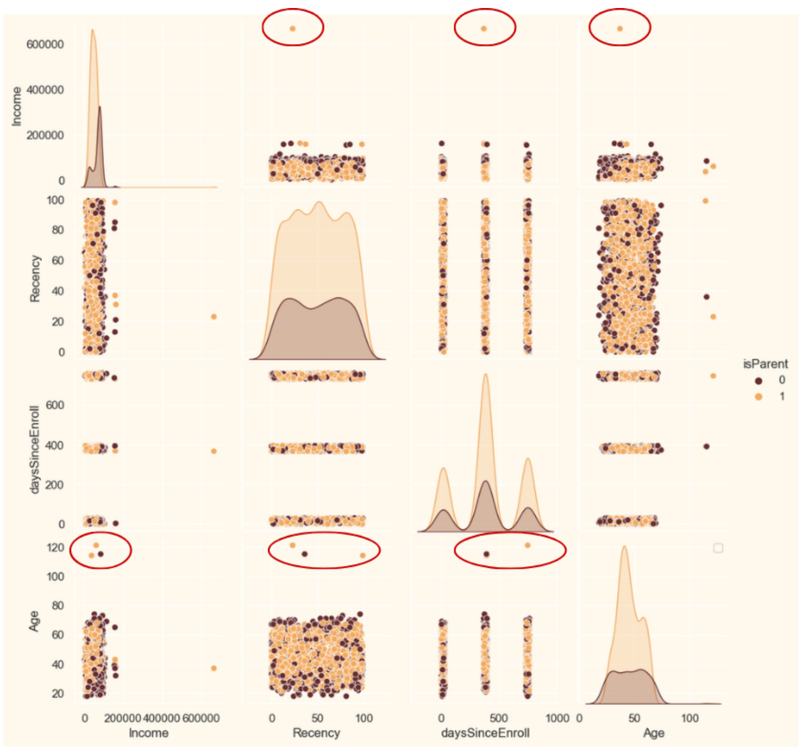


Figure 2: Pairplot to detect outliers. Outliers (circled) exist for *Age* and *Income*

3 Exploratory Data Analysis

Now that we have cleaned the data, we perform some exploratory data analysis and see if there are any interesting patterns.

We first explore the feature distribution of our dataset. From Figure 3 we can observe that our dataset contains customers from around age 20 to age 80, and we recorded customers with different financial abilities. Furthermore, our dataset also records customers' purchasing tendencies, where we recorded the customers' number of purchases via either online or in store.

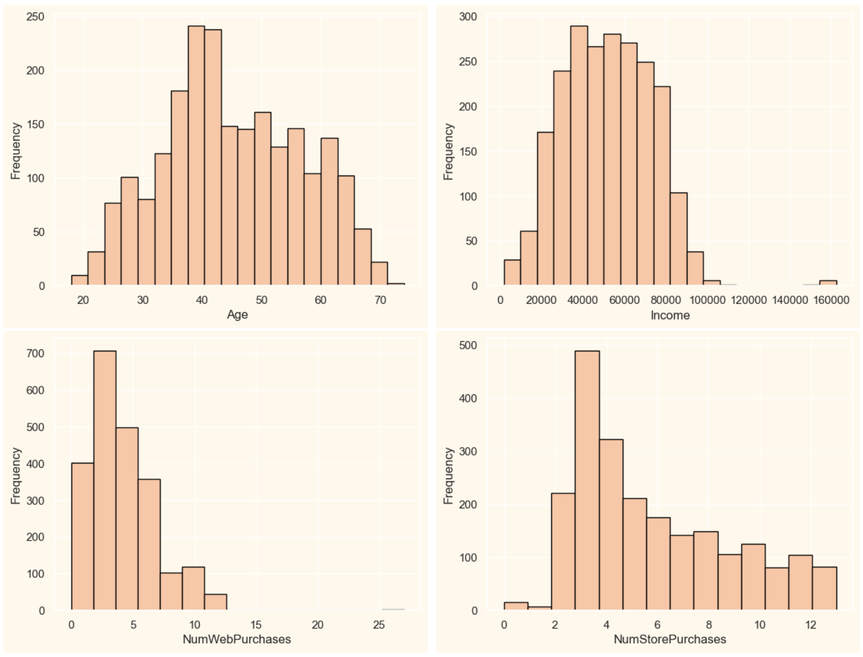


Figure 3: Data frequencies of Age, Income, WebPurchases and StorePurchases.

3.1 Relationship of *Age* with other features

We also try to analyze our dataset on bivariate relationships, and we start with the two features *Age* and *Expense*. From Figure 4 we can observe that the customers in our dataset ages from 18 to 74, and while the expenses of each age would be around one thousand, the maximum expense observed is from customer with age 73.

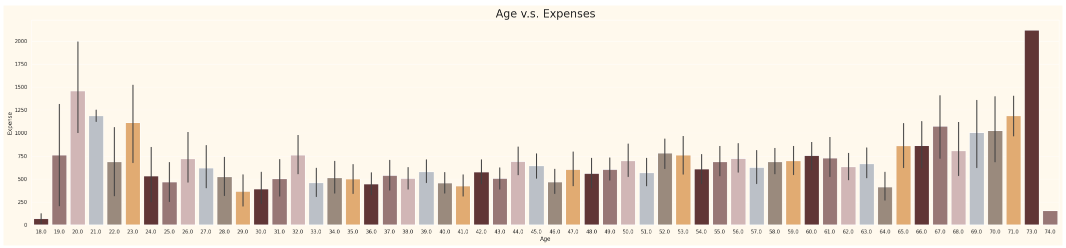


Figure 4: Barplot for Age the Expense.

In Figure 5 we continue to explore if the age of customers would affect the number of gold accessory product that they would buy. And from the barplot we can observe that there's actually no obvious trend that as person with certain age would buy more gold products.

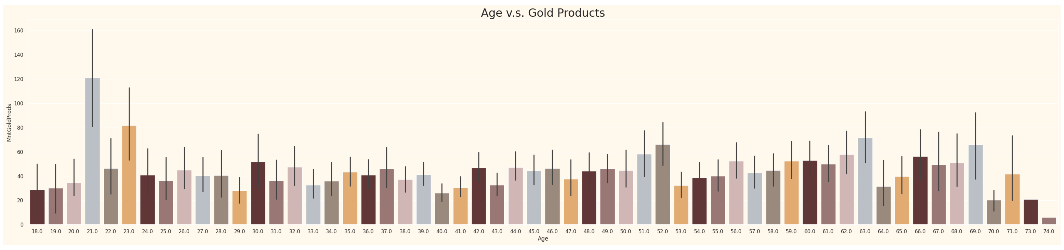


Figure 5: Barplot for Age and Gold.

In Figure 6 we try to find out if the age of customer would affect their interest in joining campaigns. From the plot we can somehow observe that customer with higher ages or lower ages would have higher respondent to the sales campaign.

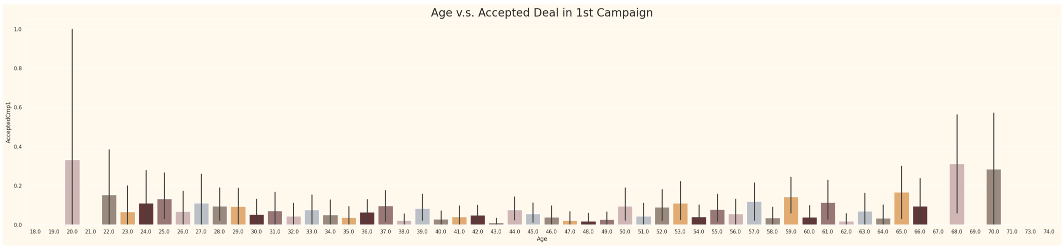


Figure 6: Barplot for Age and Acceptance to the 1st Campaign.

3.2 Relationship of *isParent* with other features

In Figure 7 we also try to explore interesting patterns such as the relationship between the customers' parential status and the number of wines being bought by the customers. And we can see from the plot, customers that are not parents on average would buy more wines than those who are parents.

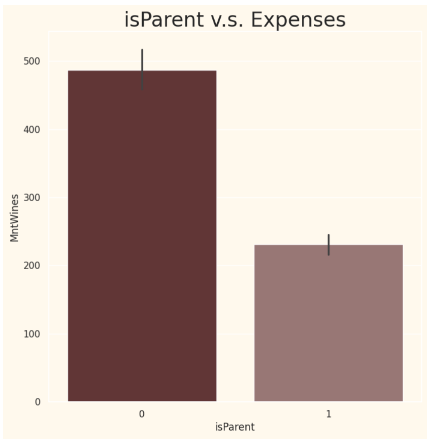


Figure 7: Barplot for isParent and MntWines.

3.3 Relationship of *Education* with other features

In Figure 8 we also try to explore interesting patterns for the feature *Education_id*. In particular we look into its relationship with campaign participation and web purchasing, and from the plots we may find out that customers with PhD education status are more likely to respond to campaigns and buy products via the web, while customers with undergraduate and master education status have similar tendency of responding to campaigns and buying products via the web.

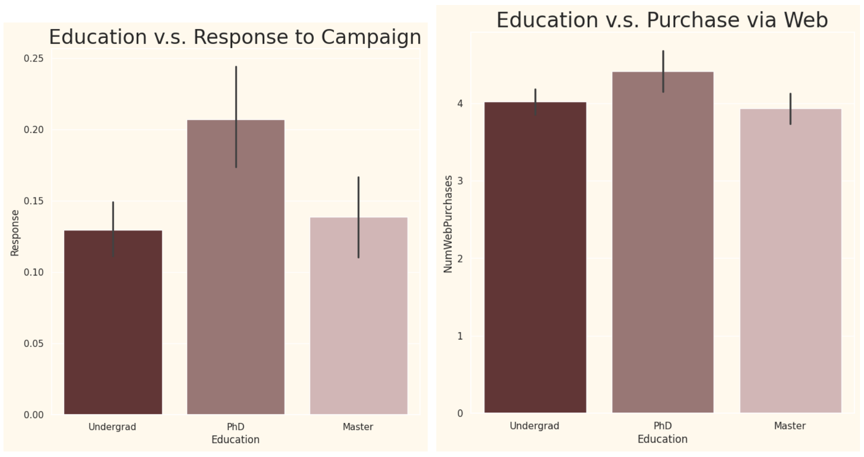


Figure 8: Barplot for Campaign Participation (left) and Web Purchasing (right).

4 Linear Regression Analysis

To answer the first question proposed in Section 1, we conduct linear regression using demographic information to predict the overall expenses.

Before fitting the model, we need to perform a series of standard data transformations. The first step is to convert strings in categorical columns to indices. To this end, we employ the `StringIndexer` class from the Spark MLlib to transform the *Education* and *Marital_Status* columns. After string indexing, the Education column takes value in $\{0, 1, 2\}$ and the *Marital_Status* column takes value in $\{0, 1\}$. In the next step, we apply one-hot encoding for all the categorical columns. The tool we adopt is the `OneHotEncoder` class in the MLlib. Now that we have all the features in numerical form, we need to assemble them into a vector for each data entry. For this process, we use the `VectorAssembler` again from the MLlib. So far, we have prepared a dense feature vector as the predictor variables, together with the *Expense* as the response variable.

Now we split all data in hand by a 7/3 ratio to form a training set and a testing set. That is, we use 70% of the data to fit the regression model and 30% to evaluate the model. Then, we standardize both the training and testing data using the statistics of the training set only, which is the standard practice to prevent data snooping. Now the data is ready for model fitting.

We fit our linear regression model using the `LinearRegression` class from the MLlib. Specifically, we set the maximum iteration to be 200, the regularization parameter to be 0.1, and the elastic net parameter to be 0.5. The regression results are reported in Table 2. The R^2 statistics of the fit is 0.6920, indicating that around 69% of the variability in the data is explained by the model. We can also see in Figure 9 that there exist no clear pattern in the residuals, meaning that our model provides an appropriate fit to the data.

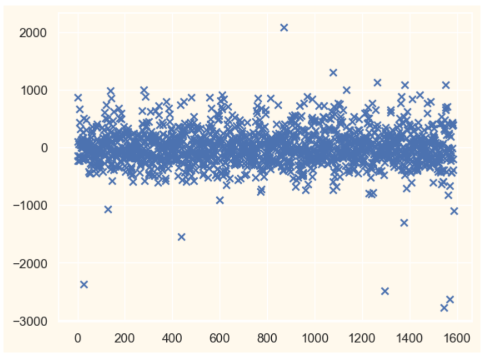
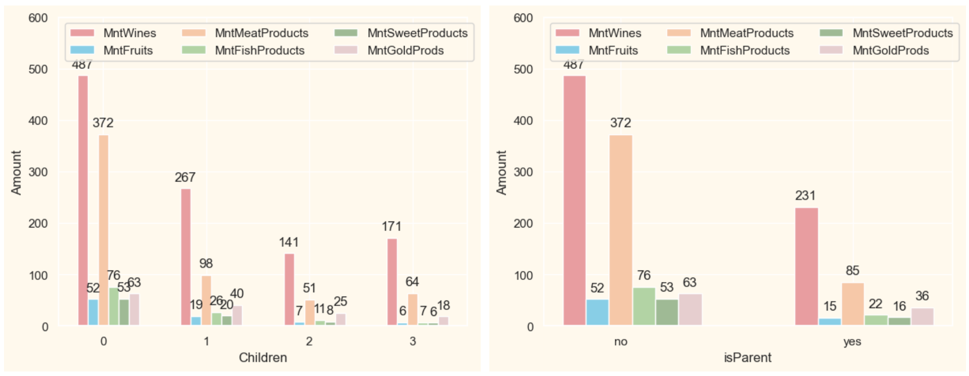


Figure 9: Residual plot.

Variable	Coefficient
isParent_0	29.2419
isParent_1	-29.2419
Education_0	7.2467
Education_1	0.0
Education_2	-3.6469
Marital_Status_0	0.2004
Marital_Status_0	-0.2004
Income	403.0858
Recency	0.6839
Age	-6.7880
Children	-114.8630
daysSinceEnroll	-94.6638

Table 2: Linear regression coefficients.

We can see from the table that the most influential predictor is *Income*, which is positively correlated with *Expense*. The top 5 most important predictors also include {*Children*: -114.86, *daysSinceEnroll*: -94.66, *isParent*=no: 29.24, *isParent*=yes: -29.24}. In order to consolidate such findings, we use several barplots to help visualize the correlations. As displayed in Figure 10, indeed, the more children the customer has, the less product he or she is likely to buy. This pattern agrees with our regression results.



(a) Amount v.s. Children (b) Amount v.s. isParent

Figure 10: Barplots to validate regression findings. Each color corresponds to one type of product.

Finally, we make predictions using the fitted model on test data. The R^2 on the test set is 0.7541, which means our model does not suffer from overfitting and performs well enough on test data. Thus, in Section 6, we will use it for streaming data prediction as more customer records come in.

5 Clustering and Customer Profiling

In this section, we also conducted K-means clustering algorithm on our dataset and try to get some insights from our results.

5.1 Clustering

For this part, we make use of MLlib for most of our algorithms, and to fit our dataset into the library's standard, we would need to transform our dataset first. We first drop certain columns that are not related to customers' personal information, such as campaign participation history (*AcceptCmp1, AcceptCmp2, AcceptCmp3, AcceptCmp4, AcceptCmp5*), complaining history and response history. Then, we transform all our features into a dense vector with *VectorAssembler* of MLlib, and then we stardardlize our features with the *StandardScaler* from MLlib. Now, we have had our dataset ready for further MLlib algorithms.

As we would like to visualize our data to get a better insights of our clustering, the dimension of our dense feature vector would be hard for us to visualize and understand. Thus, we would like to first reduce the dimension of our dataset, and we make use of the *PCA* algorithm from MLlib for this purpose. We reduce the dimension of our dataset to 3 with the Principal Components Analysis algorithm, and with the plot in Figure 11 we can now have brief understanding of our data distribution.

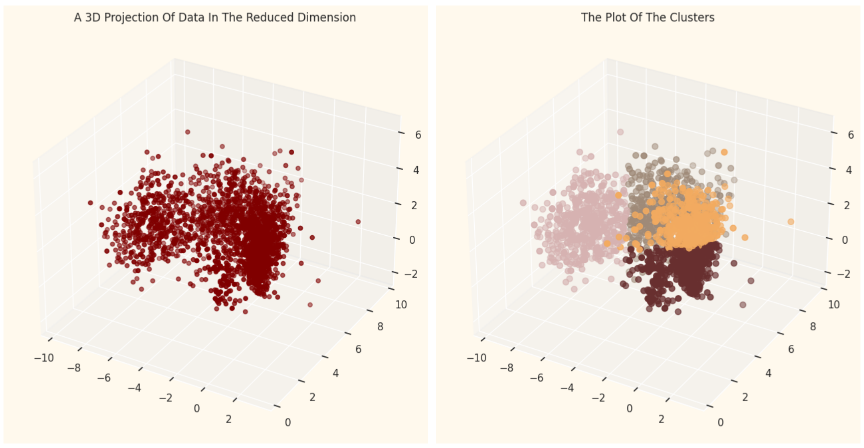


Figure 11: 3D plot of our PCA transformed dataset (left) and our clustering outcomes (right).

After performing dimension reduction with Principal Component Analysis, we are now ready to perform k-means clustering. We make use of the *KMeans* algorithm from MLlib and cluster our PCA transformed data into 4 clusters. Then, we plot the clustered results as the right plot in Figure 11, and we can clearly observe the distribution of the 4 clusters in the figure.

5.2 Profiling

From the above section we can clearly observe the distribution of our 4 clusters from Figure 11. However, as we are clustering on the PCA transformed data, we actually have no idea about the relationship between our clusters and our actual features. Thus, in this section we will do some profiling and try to explore the substantial meaning of our clusters.

We first start with the number of examples for each of our clusters, and from Figure 12 we can clearly observe that the clusters' number of examples are actually quite balanced.

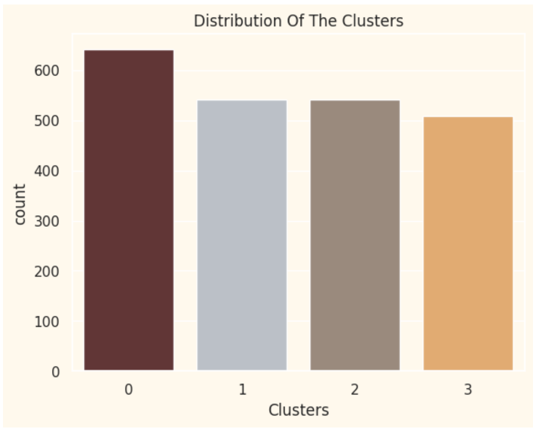


Figure 12: Number of examples of each of our clusters.

We then go on to explore the clusters' relationship with different features, and to explore on the relationship with each customer's total expenditure, we first add a column *Spending*, which is the sum of values of the columns *MntWines, MntFruits, MntMeatProducts, MntMeatProducts, MntFishProducts, MntSweetProducts* and *MntGoldProds*. We then plot the clustering results on a scatter plot of *Income* and *Spending* as shown in Figure 13. We can clearly observe the clusters' relationship with the two features. From the figure, we can conclude that examples clustered to cluster 1 would generally have higher income and higher spending budgets, while examples in cluster 0 would feature lower income and lower overall expenditure.

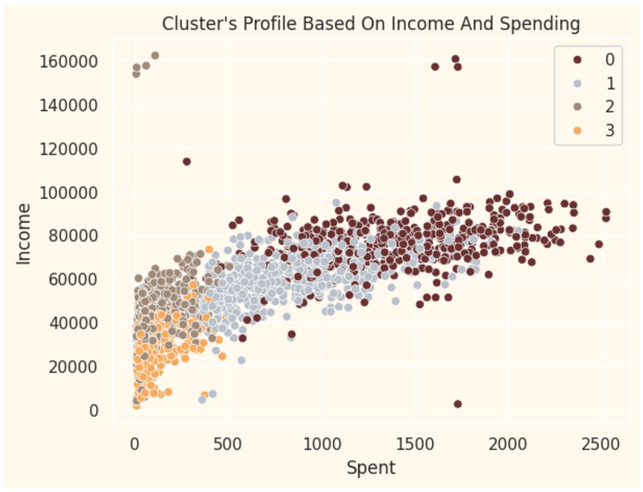


Figure 13: Cluster's Profile Based On Income And Spending

We also tried to explore the clusters' relationship between *Age* and *daysSinceEnroll*, and we plot the joint-plots in Figure 14. From the plots we can actually conclude that these two feature may have less decisive power on the clustering process, since the 4 clusters we get from the k-means clustering algorithm actually spans through most of the values of *Age* and *daysSinceEnroll*, and we cannot observe clear relations between our clusters and the two features.

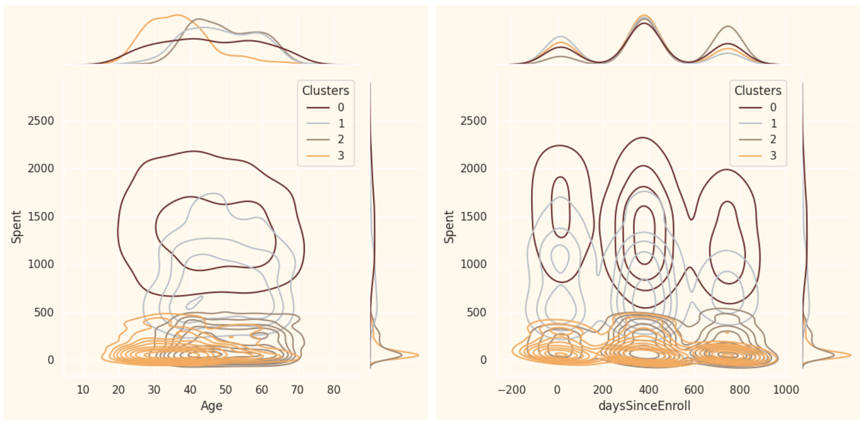


Figure 14: Cluster's Profile Based On Income And Age (left) And Enrollment Periods (right).

We then go on to explore some family status related feature: *Children* and *isParent*. From Figure 15 we can clearly find out that customers in cluster 0 tends to have lower numbers of children, while other cluster that normally have lower expenditure would have more children. One possible explanation to this phenomenon may be that the family budgets may be splitted with the children's education and healthcare, and such customers would have lower budget for groceries. The similar cluster distribution can also be observe in the right plot of Figure 15, which makes sense since the two feature *Children* and *isParent* would normally have high correlations.

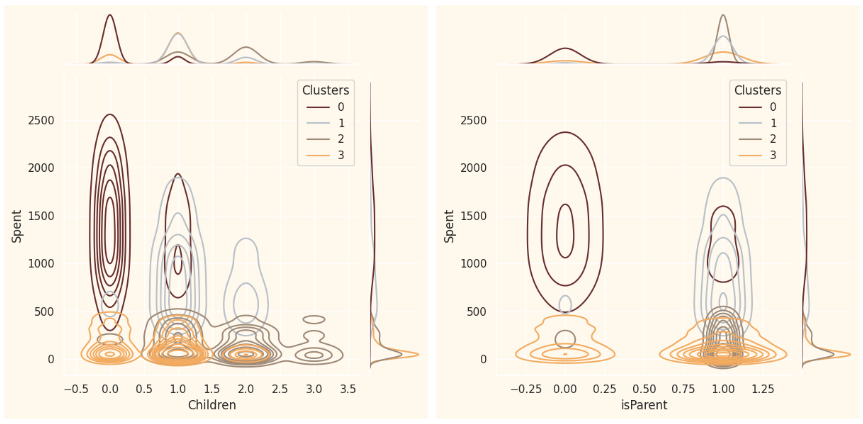


Figure 15: Cluster's Profile Based On Income And Children Numbers (left) And Parent Status (right).

Lastly, we would also like to visualize the clusters' relationship with some personal status, and specifically we plot our cluster results against *Education_id* and *Marital_Status_id*. From the left plot of Figure 16 we can observe that the 4 clusters normally have denser distribution where *Education_id* is Bachelor, and this actually make sense since normally there will be less people when the education level higher. Other than the above observed distribution, the 4 cluster actually spans through different levels in *Education_id* and we may conclude that the feature may have less power when it come to our clustering decisions. Similarly, we may also make similar conclusion for *Marital_Status_id*.

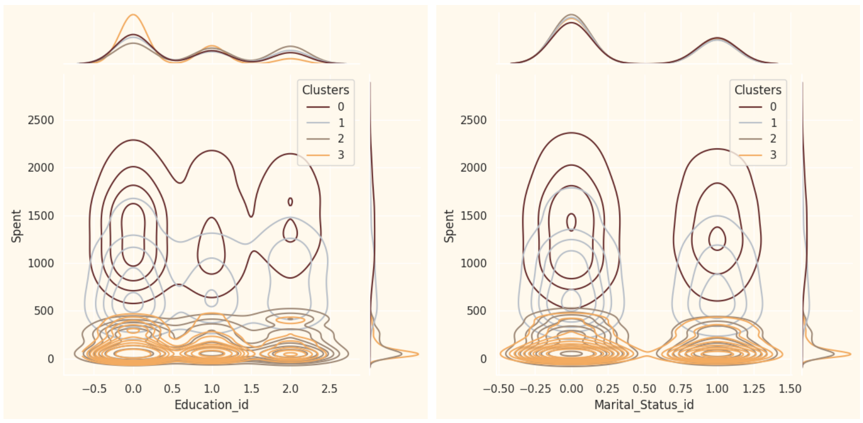


Figure 16: Cluster's Profile Based On Income And Education Status (left) And Marital Status (right).

6 Data Streaming

In this section, we build a streaming prototype to simulate the scenario where new customer records dynamically come into our database and our job is to estimate their expenses.

The input data stream comes from a csv file that contains several entries for demonstration. In practice, we would assume that input data are cleaned first (following the procedure in Section 2), and either stored in the csv file or delivered to the driver program via a port. The batch interval is set to 2 seconds.

To start with, we first read the streaming data as an RDD (as shown in Figure 17), and manually convert them to a dataframe to apply data processing operations. As opposed to directly reading the data in as a dataframe in structure streaming, we choose this approach because the operations to apply on the dataframe is highly user-defined and non-generic, such as string indexing and one-hot encoding. Consequently, when we first tried to implement it using structure streaming, it just wouldn't work. We hope to see future updates of structured streaming to support more user-defined behaviors on streaming dataframes.

```python
from itertools import islice

def split(s):
    kvs = s.split(',')
    vs = []
    for item in kvs:
        if item.isdigit():
            vs.append(int(item))
        else:
            vs.append(item)
    return vs

rdd = sc.textFile('../stream_data/stream.csv')
rdd.cache()

rdd = rdd.mapPartitionsWithIndex(lambda idx, it: islice(it, 1, None) if idx == 0 else it)
rdd = rdd.map(split)
rdd.collect()
```

Figure 17: Code snippet of reading streaming data as an RDD, instead of a dataframe.

To transform data entries to dense vectors, we implement a `feature_builder()` function to perform string indexing, one-hot encoding, and vector assembling. Notably, we added several dummy entries in the one-hot encoding step so as to facilitate the error-free execution of the one-hot encoder. Although this does not seem to hurt performance, we hope to optimize this part of code in the future. Then, we convert the processed dataframe to an RDD and apply a transform function to use the linear model for prediction. The predictions are then printed using a customized print function.

7 Conclusion and Future Work

In this project we make use of the Customer Personality Analysis dataset from Kaggle dataset to perform a sequence task of data cleansing, data pre-processing and exploratory data analysis. We also further make use of the MLlib to explore the possible solutions with linear regression and k-means clustering, and we also conducted profiling to link the clustering results with our data. Last but not the least, we constructed a data streaming prototype where we can simulate the dynamic data receiving.

As for future work, there are several features that can be further developed. First, apart from simply making inference with the linear regression model that is fitted on static data, we may support continued training of the model whenever more streaming data, which are associated with ground truth expense values, come in. This strategy will keep the model up to date and provide more accurate predictions. Second, we can try to optimize the codes for streaming function with structured streaming so that fewer low-level operations need to be explicitly written.

8 Declaration of the use of AI tools

ChatGPT was used for polishing the introduction of this report.

References

[1] Matplotlib. URL: https://matplotlib.org/.

[2] Spark MLlib. URL: https://spark.apache.org/docs/latest/ml-guide.

[3] Pandas. URL: https://pandas.pydata.org/.

[4] PySpark. URL: https://spark.apache.org/docs/latest/api/python/index.html.

[5] Spark RDD. URL: https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.html.

[6] Spark SQL. URL: https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/index.html.